The Effect of Population Size on the Fitness Scores of the NEAT Algorithm

Natasha Sitsky

Aim

To determine the effect of the population size of cars on their best fitness scores after 5 and 10 generations.

Hypothesis

If the best fitness achieved by the NEAT algorithm is affected by the population size, then as the population size increases, the best fitness achieved will also increase. This is because a larger population size will allow for more random mutations, which increases the likelihood of a successful mutation taking place.

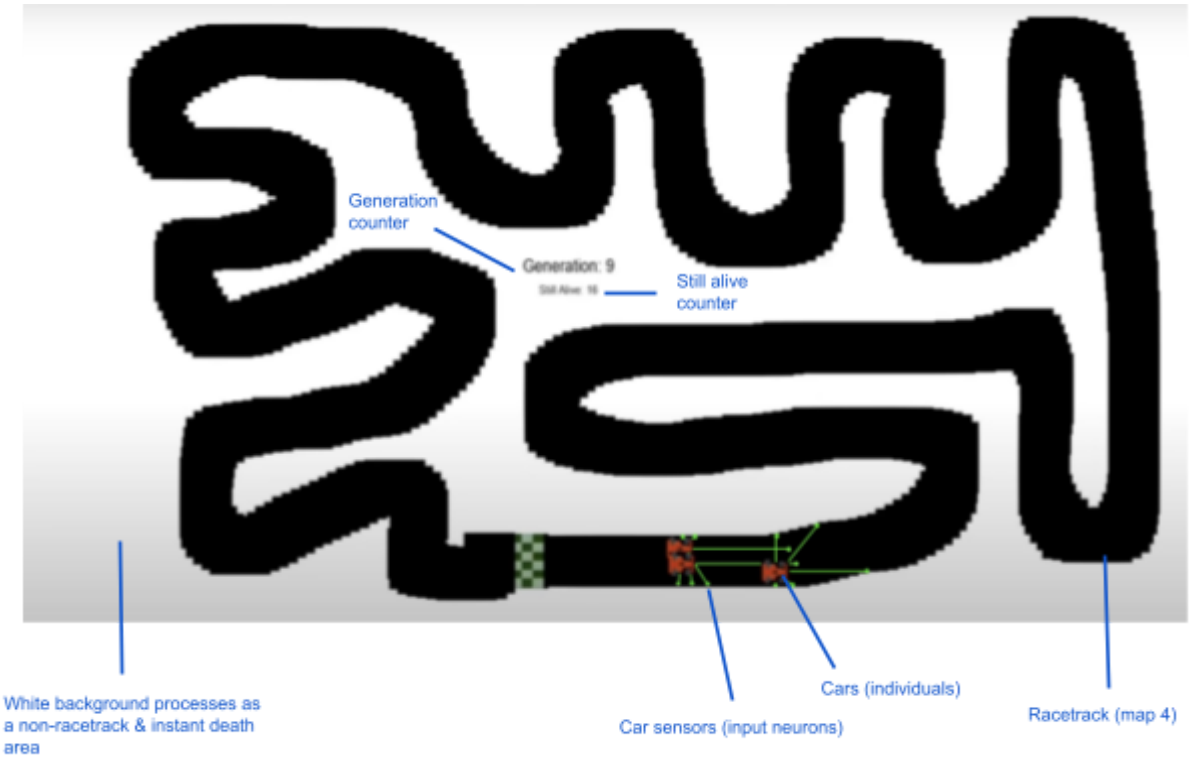
Variables

Independent variable: Population size. This refers to the number of cars which start at the beginning of each generation. It is defined on line 4 of config.txt through the variable pop_size.

Dependent variable: Highest fitness score after 5 & 10 generations. This can be measured from the program's records which are printed in the terminal as the experiment progresses.

Controlled Variables		**Why does it need to be controlled?**	**How will it be controlled?**
Variables defined in config.txt	Activation function (tanh)	Different activation can produce varying outputs as they impact which neurons should fire based on input data, and each have their own specifics and issues, etc.	The activation function will not be edited in config.txt during the experiment.
	Number of elitists (3)	Elitists are the individuals that continue to the next generation unchanged. Therefore a differing number of elitists will impact how the algorithm progresses.	The number of elitists will not be changed in config.txt during the experiment.
	Other variables	Each variable will slightly differ how the program runs, therefore, they must all be kept constant to ensure a consistent testing environment.	Config.txt will not be edited or changed during the experiment.
Other	Chosen map (map 4)	Different maps have different difficulties, which will impact how hard it is for cars to progress. This will impact their overall fitness as a harder map will not allow them to get as far.	The map chosen will not be edited in Newcar.py throughout the experiment.
	Computer used	Different computers have different processing speeds which could impact how quickly / well the program runs.	I will only use my laptop to run the experiment.
	Background programs running	Running a large amount of background programs during the experiment could potentially slow or impact how the program runs.	I will keep the same amount of background programs running throughout the experiment.

Diagram



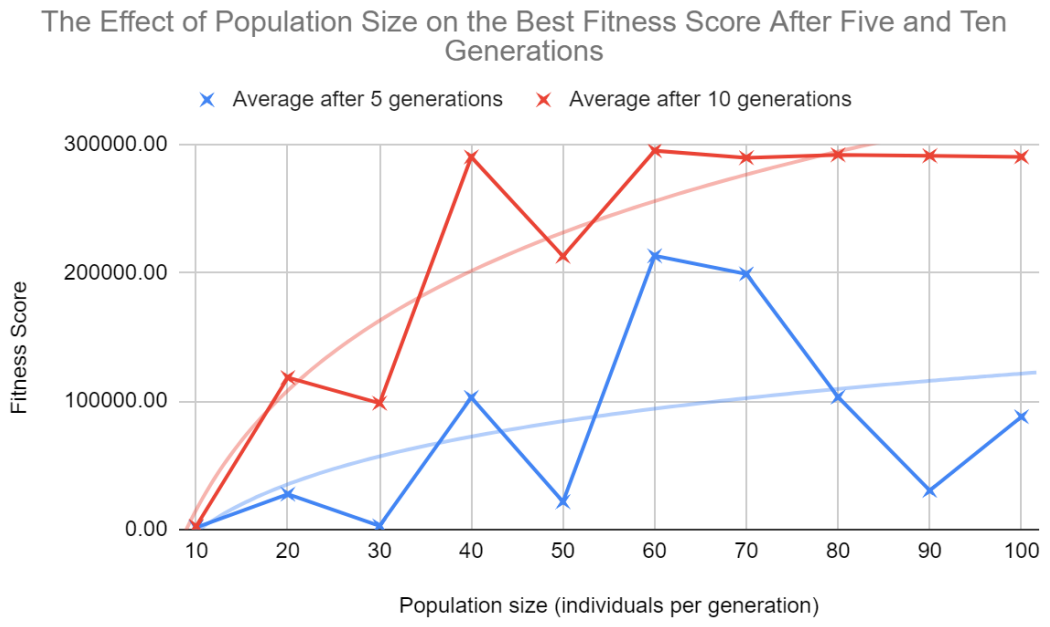
Method

1) Download the code from the jetscholar's ML_Task2 GitHub repository using the command `git clone https://github.com/jetscholar/ML_Task2.git`
2) Open the code in Visual Studios.
3) Open config.txt and change the population size (pop_size) to 10.
4) Save the file and enter `python newcar.py` into the terminal to run the code.
5) Run the simulation for 10 generations, and then press Alt-F4 to stop the program.
6) Record the best fitness score achieved after 5 and 10 generations from the terminal, where it will have been recorded automatically by the program.
7) Repeat steps five to six twice more in order to have three trials.
8) Repeat steps three to seven until you have trialled population values 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100.

Results

The Effect of Population Size on the Best Fitness Score After Five and Ten Generations

Population size	Best fitness achieved after 5 generations				Best fitness achieved after 10 generations			
	Trial 1	Trial 2	Trial 3	Average	Trial 1	Trial 2	Trial 3	Average
10	3979.20	305.20	771.33	1685.24	3979.20	424.67	3289.53	2564.47
20	18138.67	2514.67	62080.00	27577.78	290315.07	2605.20	62080.00	118333.42
30	3081.87	1911.33	3761.33	2918.18	3081.87	3079.53	289039.33	98400.24
40	2838.53	16606.40	289039.33	102828.09	289039.33	290951.33	290235.33	290075.33
50	55930.00	2470.00	6562.67	21654.22	291030.67	58065.33	289358.67	212818.22
60	190376.07	159547.73	289358.67	213094.16	304866.47	290076.07	289987.20	294976.58
70	18018.40	289358.67	289677.73	199018.27	289358.67	289358.67	289677.73	289465.02
80	3283.73	17187.73	289039.33	103170.27	289358.67	289757.33	296220.07	291778.69
90	18018.40	54993.13	18217.73	30409.76	289597.87	291269.07	292220.67	291029.20
100	56091.67	17398.53	189989.67	87826.62	289358.67	289039.33	292161.60	290186.53



Discussion

The trend shown in this data for the 10th generation clearly shows that a higher population size does increase the best fitness results. This proves my hypothesis correct. The trend can be explained by the same reasoning seen in my hypothesis: a larger population size allows for more random mutations to take place, thus likening the chance of a more successful mutation. A more successful mutation is able to travel further, and will breed with other successful cars in order to further the development of the best fitness values. However, these results also slightly differ from my hypothesis, as after about 60 cars per generation, the effect essentially stops and all population sizes higher have around the same fitness values. This showed a logarithmic trend rather than linear or exponential one which I expected. The results after five generations did not show the trend nearly as well, likely because there is not enough time for the algorithm to properly develop. The total randomness of the first generation means that there is likely to be some variation in the results, especially in the earlier generations, which have not had enough time to mutate new and better combinations.

Although the experiment shows logical and clear results, there are several improvements that could be made, which would improve the accuracy of the data and overall trendline. As discussed above, the fitness data after 5 generations showed a much less coherent trend than the data recorded after 10 generations. This is because the program has not had enough time to properly execute the NEAT algorithm to its full potential. Despite the obvious improvement of the 10th generation data over the 5th, the data points are still a little uneven when compared to the trendline, which is a reflection of the randomness of the algorithm. To improve this, testing could be done for a longer period of time, such as 20 or 30 generations, which would allow the randomness of the simulation to even out. In this experiment, I simply could not test for longer due to time constraints, but a later exploration may not have this issue. This issue could also be resolved by running multiple simulations simultaneously, which would require some amount of organisation and structure, but could significantly cut down on testing time required. A further way to reduce the impact of randomness on the data is to complete more trials. This could be 5 or even 10+ trials in order to get the most accurate averages possible. A further test aimed more specifically at the generality of the most successful cars could also look to making use of randomising the maps every generation.

Conclusion

In summary, a larger population size of the cars in this NEAT simulation leads to a higher best fitness value. This effect is seen in a logarithmic trend, which has little to no difference for populations of 60 or higher. Despite recording results at both the 5th and 10th generation, future experiments would likely need even later generations and more trials to show the trend in a clearer manner, and decide on the exact differences between the population values of 60+.

Bibliography

Code is heavily inspired by the Youtuber Cheesy AI, and was edited & commented by NeuralNine (Florian Dedov), before being forked from the jetscholar ML_Task2 repository.

ChatGPT, 2023 (used to help understand certain sections of the code when commenting)

Baheti, P. (2021) *Activation functions in neural networks [12 types & use cases]*, *V7*. Available at: https://www.v7labs.com/blog/neural-networks-activation-functions#:~:text=An%20Activation%20Function%20decides%20whether,prediction%20using%20simpler%20mathematical%20operations.

Das, S. and Panigrahi, B.K. (2009) *What is Elitism*, *IGI Global*. Available at: https://www.igi-global.com/dictionary/multi-objective-evolutionary-algorithms/9592.

Neat overview (2019) *NEAT*. Available at: https://neat-python.readthedocs.io/en/latest/neat_overview.html.

NeuralNine. (2021) Self-Driving AI Car Simulation in Python. [Online]. Available at https://www.youtube.com/watch?v=Cy155O5R1Oo

Shorten C. (2019) *Neuroevolution of Augmenting Topologies (NEAT)*. [Online]. Available at https://www.youtube.com/watch?v=b3D8jPmcw-g&t=635s